Font Manager

Session 201

Font Manager

Xavier Legros
Mac OS X Evangelist
Apple Worldwide Developer Relations

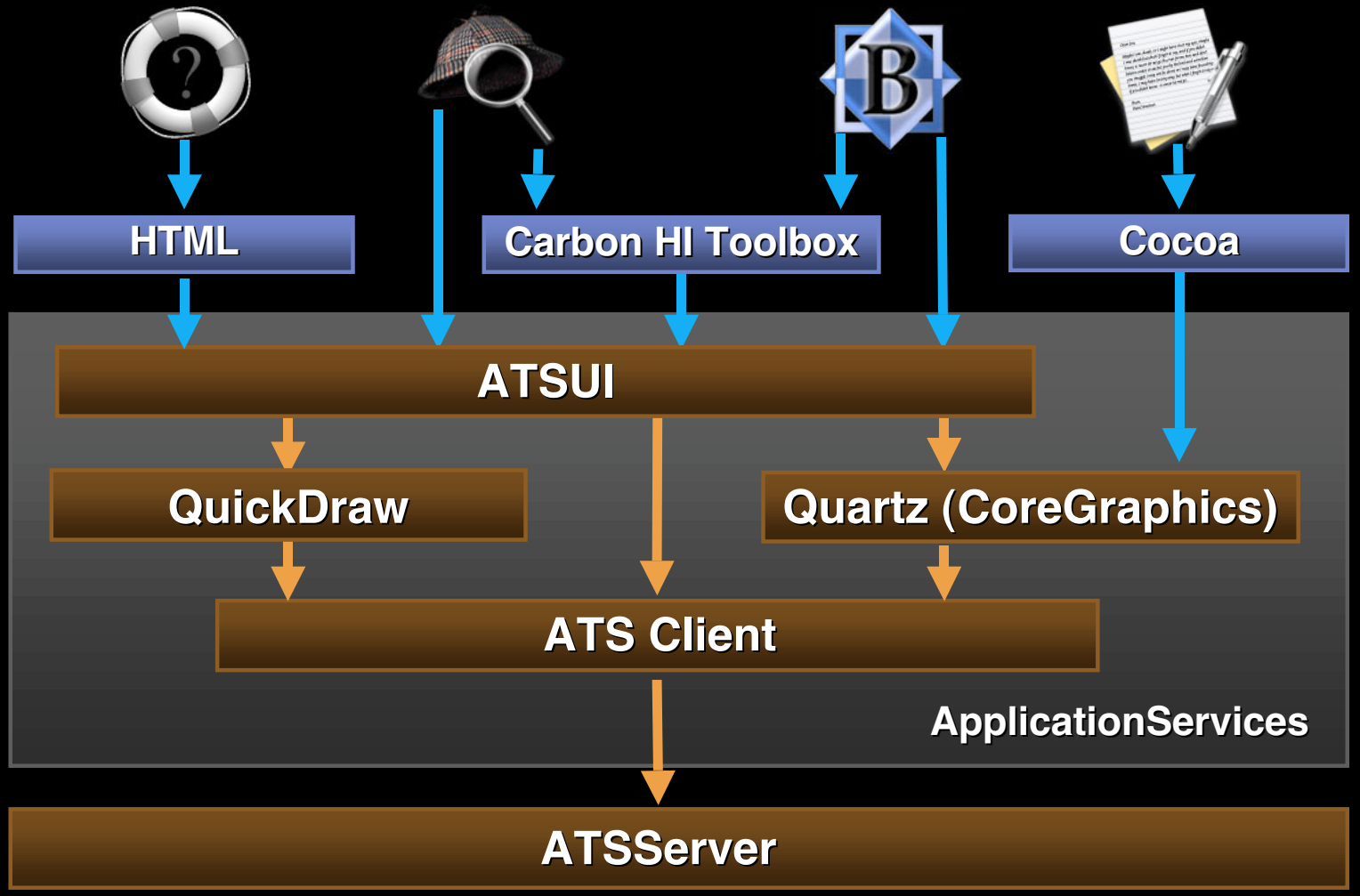
Agenda

- Quick overview
- Features and enhancements for Jaguar
- Demonstration
- Fonts programming interface

Font Manager

Mike Conley
Senior Software Engineer, Type Group

Mac OS X Font Services



Font Formats

- Macintosh TrueType font suitcase
- Windows TrueType (.ttf/.ttc)
- PostScript OpenType, OpenType CID, and Macintosh Type 1 (LWFN, SFNT, SFNT/CID)

Font Formats

- Macintosh TrueType font suitcase
- Windows TrueType (.ttf/.ttc)
- PostScript OpenType, OpenType CID, and Macintosh Type 1 (LWFN, SFNT, SFNT/CID)
- PostScript Multiple Master support
- Enhanced OpenType support
- Protected CID support

Standard User Font Locations

- User: ~/Library/Fonts/
- Local: /Library/Fonts/
- Network: /Network/Library/Fonts/
- System: /System/Library/Fonts/
- Classic: System Folder:Fonts:

Programmatic Font Activation

- From any directory, including application library directory and application bundle
- From application resource fork

Programmatic Font Activation

- From any directory, including application library directory and application bundle
- From application resource fork
- Automatic font activation

Installing Fonts

- Drag and drop to font directory
- Directories scanned on application launch and user login
- Resolution of duplicate and conflicting fonts

Installing Fonts

- Drag and drop to font directory
- Directories scanned on application launch and user login
- Resolution of duplicate and conflicting fonts
- Nested font subdirectories
- Font notification

Font Selection

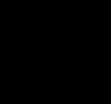
- Font menu for Carbon apps
- Standardized appearance via API

Font Selection

- Font menu for Carbon apps
- Standardized appearance via API
- Font Panel for Carbon apps

Jaguar Features and Enhancements

- Font formats
- Nested font subdirectories
- Font notification
- Font auto-activation
- Font Panel for Carbon
- Improved stability
- Enhanced performance



PostScript Multiple-Master Fonts

- PostScript Type 1 Multiple-Master fonts
- Packaged as bitmap font suitcases paired with outline font files

Enhanced OpenType Support

- Synthesizes ‘FOND’ data identical to ATM’s
- Fully compatible kerning and font metrics
- Minimal code changes required
- On-demand synthesis and caching

Protected CID Support

- Copy-protected CJKV PostScript fonts
- Used extensively in Asian business and design/publishing professionals
- Works for CID fonts that are already installed in Classic font folder

Nested Font Subdirectories

- Drag and drop to font directory
- Drop entire font folders

Font Notification

- Notification of font activation and deactivation
- Requires CFRunLoop-based client
- Subscribe and unsubscribe to notification
- Servers and tools can get immediate notification

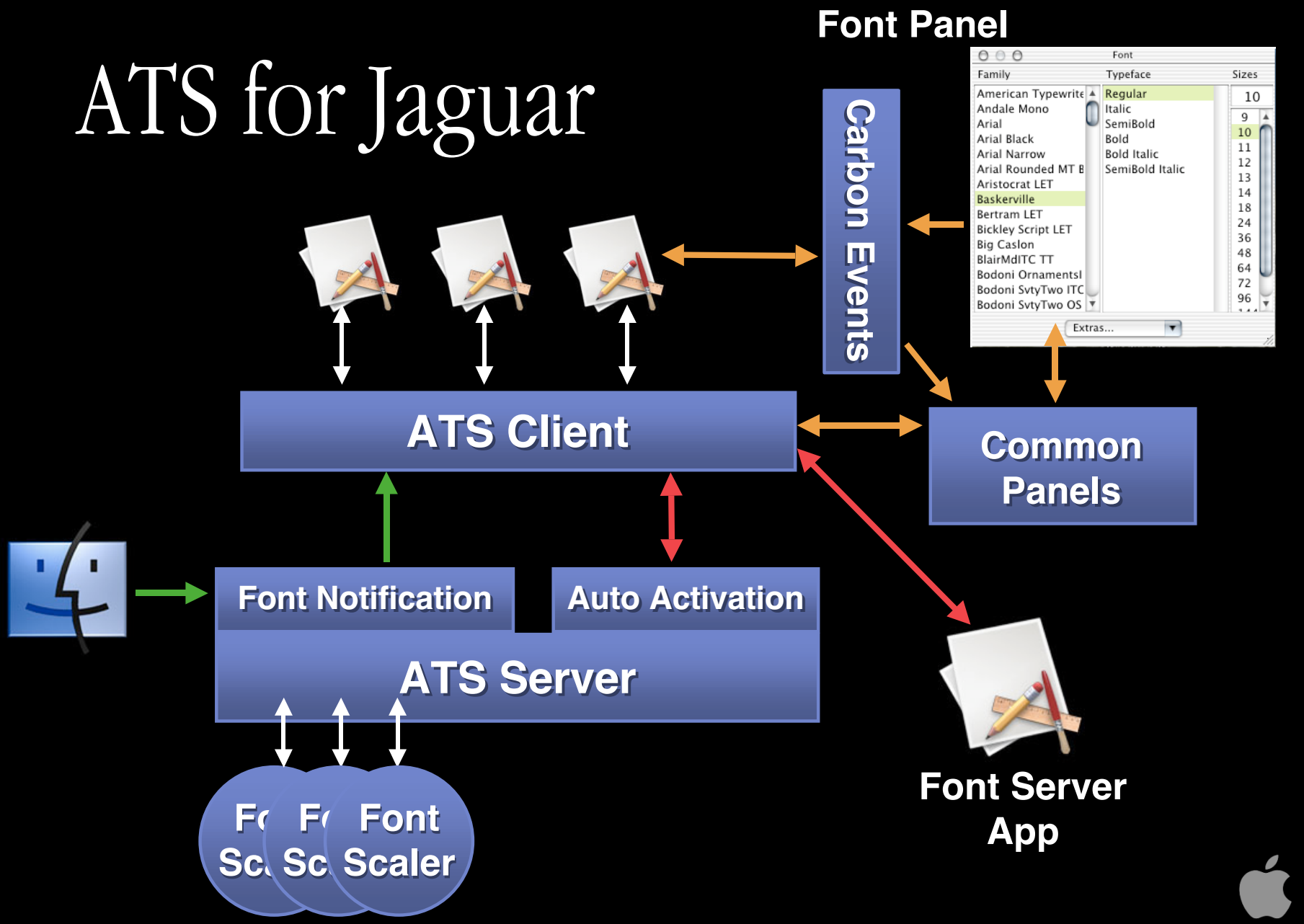
Font Auto-Activation

- Clients register auto-activation port
- ATS calls clients when it cannot locate a font
- Client activates font and returns relevant data to ATS

Font Panel for Carbon

- Easy to select font, style, size, and color
- Improved handling of large collections
- Uses Carbon Events, though WaitNextEvent() apps can use it, too

ATS for Jaguar



Improved Stability

- Improved exception handling in ATS Server and font scalers
- Disabling of damaged or invalid font files
- API is now thread safe
- Better error checking
- Many bug fixes

Enhanced Performance

- Better font streaming
- Login, logout speed-up
- TrueType font scaler
- More caching
- More efficient memory allocation

Demonstration

- Font panel
- Notifications
- Auto activation

Demo

Nathan Taylor
Software Engineer, Type Group

Programming Interface

Nathan Taylor
Software Engineer, Type Group

Programming Interface Topics

- API Sets
- Programming tasks
- Performance

API Sets

- Font Manager (FM…)
  - E.g., FMCreateFontFamilyIterator
  - Available on Mac OS 9, CarbonLib, and Mac OS X
- ATS (ATS…)
  - E.g., ATSFontIteratorCreate
  - Available on Mac OS X

Font Access Data Types

- Font references
    - ATSFontRef
    - FMFont
    - ATSUFontID
- Family references
    - ATSFontFamilyRef
    - FMFontFamily
- These can be different!

Programming Tasks

- Activation and enumeration
- Storing references
- Conversion and compatibility
- Notification
- Font panel and menu
- Auto activation

Task: Activation

- Activation
    - ATSFontActivateFromFileSpecification
    - ATSFontActivateFromMemory
    - FMActivateFonts
- Deactivation
    - ATSFontDeactivate
    - FMDeactivateFonts

Closeup: Activation

```
OSStatus
ATSFontActivateFromFileSpecification (
                        FSSpec *                iFile,
                        ATSFontContext          iContext,
                        ATSFontFormat           iFormat,
                        void *                  reserved,
                        ATSOptionFlags          iOptions,
                        ATSFontContainerRef *   oContainer );
```

- iContext — font accessibility: global or local
- iOptions — used to specify fork if necessary

Example: Activation

- Activate a font locally from a file spec:

```
status = ATSFontActivateFromFileSpecification (
                        &fsSpec,
                        kATSFontContextLocal, …
                        kATSOptionFlagsDefault,
                        &container );

status = ATSFontDeactivate (
                        container, …
                        kATSOptionFlagsDefault );
```

Task: Enumeration

- Families: (Helvetica)
    - ATSFontFamilyIteratorCreate
    - FMCreateFontFamilyIterator
- Fonts: (Helvetica, Bold)
    - ATSFontIteratorCreate
    - FMCreateFontIterator
- Family instances: (Helvetica, Bold, 12 pt.)
    - FMCreateFontFamilyInstanceIterator

Closeup: Enumeration

```
OSStatus
ATSFontIteratorCreate (   ATSFontContext        iContext,
                          ATSFontFilter *       iFilter,
                          void *                iRefCon,
                          ATSOptionFlags        iOptions,
                          ATSFontIterator *     ioIterator );
```

- iContext — font accessibily: global or local
- iFilter — a caller defined filter to reduce returned fonts
- iOptions — scope details: restricted or unrestricted

Task: Enumeration

	Local Context	Global Context
Restricted Scope	Fonts activated locally to my application	Only globally activated fonts
Unrestricted Scope	Default fonts: Global and my local	ALL fonts: including other local activations

Example: Enumeration

- Create an iterator to enumerate all the fonts in the applications context:

```
status = ATSFontIteratorCreate (
                    kATSFontContextLocal, ...
                    kATSOptionFlagsUnrestrictedScope,
                    &iterator );

while ( ATSFontIteratorNext ( &iterator, &fontref ) == noErr ) {
      ...
}
```

Example: Enumeration

- Status codes
    - kATSIterationCompleted
        - Finished, nothing more to do
    - kATSIterationScopeModified
        - Something changed
        - Reset iteration and try again

Task: Notifications

- Create a callback
    - Handle the notification, do something
- Subscribe to notifications
    - Register the callback
- Notify other applications
- Unsubscribe

Example: Notification

- Create a callback to handle the notification

```
static void NotificationCallback (
                    ATSFontNotificationInfoRef      info,
                    void *                          refCon )
{
    …
    UpdateStandardFontMenu(…);
}
```

Example: Notification

- Register your callback to receive notifications

```
status = ATSFontNotificationSubscribe (
                        NotificationCallBack,
                        kATSFontNotifyOptionDefault,
                        NULL, // iRefCon
                        &notifyRef );
```

- Unsubscribe from, notifications

```
status = ATSFontNotificationUnsubscribe ( &notifyRef );
```

Task: Storing Font References

- Problem:
    - How to save a font reference?
- Good:
    - Family name with style
    - Postscript name
    - Full or Unique name
- Bad:
    - Family ID
    - Font ID

Task: Conversion

- Conversion
    - FMGetFontFamilyInstanceFromFont
    - FMGetFontFromFontFamilyInstance

Task: Compatibility

- Compatibility APIs
  - ATSFontGetFontFamilyResource
  - FMGetFontFamilyResource
  - ATSFontGetFileSpecification
  - FMGetContainerFromFontFamilyInstance
  - ATSFontFamilyFindFromQuickdrawName
  - ATSFontFamilyGetQuickdrawName

Task: Font Panel

- Use the new Font Panel
    - Toggle font panel state
    - Query font panel state
    - Update font panel display
    - Get selection
        - Carbon event suite

Example: Font Panel

- Showing and hiding the Font Panel from a carbon event handler:

```
switch ( GetEventClass ( event ) ) {
    case kEventClassCommand :
        GetEventParameter( event, …, & command );
        switch ( command.commandID ) {
            case kHICommandShowHideFontPanel:
                status = FPShowHideFontPanel ();
                break;
            …
        }
        break;
    …
}
```

Task: Font Panel

- Font panel carbon events
  - Class
    - kEventClassFont
  - Type
    - kEventFontPanelClosed
    - kEventFontSelection
  - Parameters
    - kEventParamATSUFontID, kEventParamATSUFontSize
    - kEventParamFMFontFamily, kEventParamFMFontStyle, kEventParamFMFontSize
    - kEventParamFontColor

Example: Font Panel

- Getting the font family:

```
case kEventClassFont:
    switch ( GetEventKind ( event ) ) {
        case kEventFontSelection:
            status = GetEventParameter(
                        event,
                        kEventParamFMFontFamily, …
                        &fontFamily );
            break;
    }
    break;
```

Example: Font Panel

- Update the font panel with current selection:

```
FontSelectionQDStyle  qdInfo;

qdInfo.version = kFontSelectionQDStyleVersionZero;
qdInfo.instance.fontFamily = family;
qdInfo.instance.fontStyle = style;
qdInfo.size = size;
qdInfo.hasColor = false;

status = SetFontInfoForSelection ( kFontSelectionQDType, 1,
                    &qdInfo, windowTargetRef );
```

Task: Selection

- Use the standard font menu functions
    - Create the menu
    - Update the menu
    - Determine selection

Example: Selection

- Create a basic font menu:

```
menu = GetMenuHandle(kFontMenuResID);
status = CreateStandardFontMenu(
                menu,
                CountMenuItems(menu),
                kFontMenuFirstHierMenuID,
                kHierarchicalFontMenuOption,
                &hierarchicalMenuCount);
```

Example: Selection

- Find the font family and style from the menu item selected by the user:

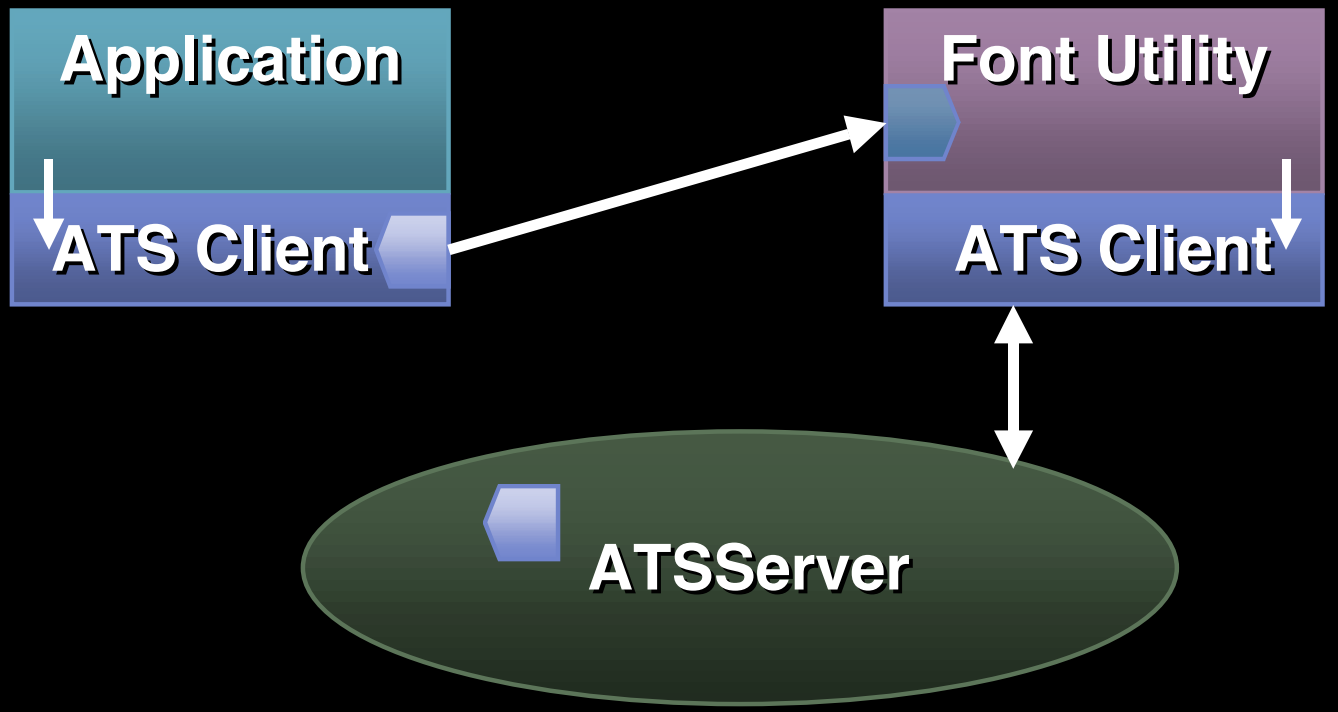
```
UpdateStandardFontMenu(GetMenuHandle(menuID),
      &hierarchicalMenuCount);

status = GetFontFamilyFromMenuSelection(
                                  GetMenuHandle(menuID),
                                  menuItem,
                                  &fontFamily,
                                  &style);
```

Task: Auto-Activation

- Create a callback to handle queries
    - kATSQueryActivateFontMessage
- Register for queries
    - ATSCreateFontQueryAndRunLoopSource
    - CFRunLoopAddSource

Task: Auto-Activation

- Auto-activation query communications



Example: Auto-Activation

- Create a callback to handle port messages

```
CFPropertyListRef
MyQueryCallBack (       ATSFontQueryMessageID       msgID,
                        CFPropertyListRef *         data,
                        void *                      refCon )
{
    switch ( msgID ) {
        case kATSFontRequestQueryMessageID:
            // Parse and handle the query
    }
}
```

Example: Auto-Activation

- Register the callback

```
CFRunLoopSourceRef          source = NULL;

source = ATSCreateFontQueryAndRunLoopSource(
                                0, 0, MyQueryCallBack, NULL );
if ( source != NULL )
    CFRunLoopAddSource (
                CFRunLoopGetCurrent (),
                source,
                kCFRunLoopDefaultMode );
```

Example: Auto-Activation

- Parse the query

```
if ( CFDictionaryContainsKey( theDict,
                kATSQueryKeyQDFamilyName ) ) {
    theName = CFDictionaryGetValue( theDict,
                            kATSQueryKeyQDFamilyName );
    … // translate theName into an FSSpec
}

status = ATSFontActivateFromFileSpecification (
                                    fontFileSpec,
                                    kATSFontContextGlobal,
                                    kATSFontFormatUnspecified,
                                    … );
```

Performance

- Use notification API
- Use performance tools
- Avoid font iteration and primitives
- Use optimized high-level functions

Summary

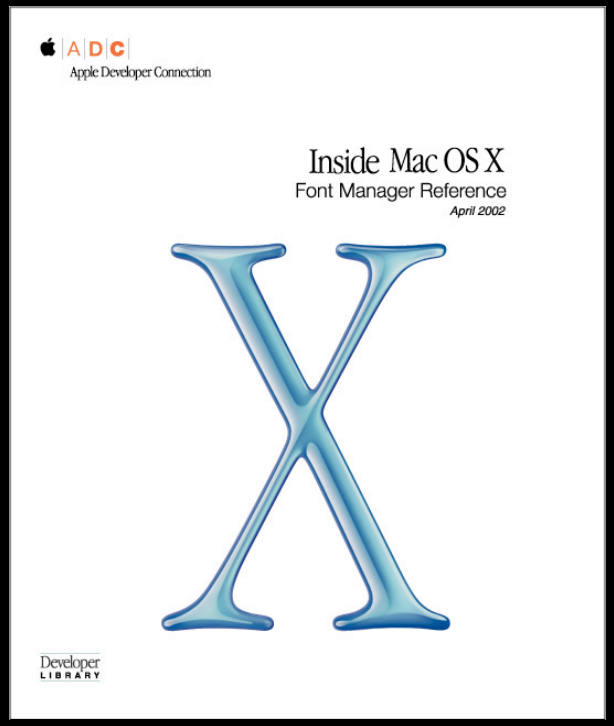
- Iterate only the fonts you really care about
- Implement the Font Panel
- Continue developing to the programming interface of Font Manager on Carbon
- Review the programming interface of ATS Framework on Mac OS X

Roadmap

200 Making Your Application Unicode Savvy	Room C **Tue., 9:00am**
202 Drawing Text With ATSUI	Room J **Tue., 3:30pm**
208 MLTE: A Unicode Text Engine	Room A2 **Thurs., 9:00am**
010 Going International With Mac OS X	Room A2 **Fri., 10:30am**

Documentation

Font Manager



- Managing Fonts With the Font Manager
- Font Manager Reference

Documentation>Carbon>Text and International Services>Font Manager
developer.apple.com/techpubs/macosx/Carbon/text/FontManager/fontmanager.html

More Documentation

Apple Type Services Technical Documentation

developer.apple.com/techpubs/macosx/Carbon/text/ATSTypes/atstypes.html

TN2024: The Mac OS X Font Manager

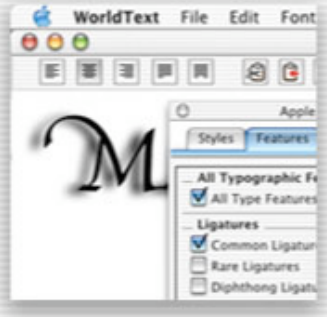
http://developer.apple.com/technotes/tn/tn2024.html

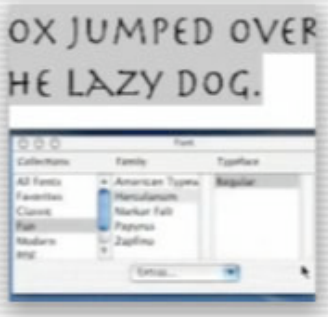
Who to Contact

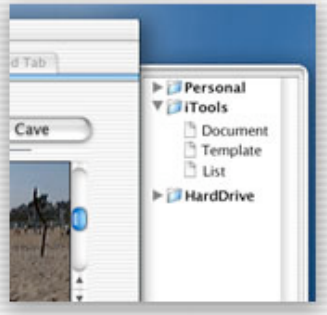
Xavier Legros
Mac OS X Evangelist
Apple Worldwide Developer Relations
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html

Q&A







Xavier Legros
Mac OS X Evangelist
xavier@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002